Assignment 3

Alanda Cherestal

November 2022

Contents

1 Introduction

In this assignment, I implemented a deterministic classifier, 1-Nearest Neighbor, to create a written digit classifier.The MNIST database of handwritten digit is given for the implementation. This assignment is written in python on Google Colab.

2 Data Processing

The dataset is prepossessed with a split of 60,000 train and 10,000 test images. Working with 60,000 training set is quite difficult, so I took a sample of 100 images from the training set. I fist defined the euclidean function to take the sum of distance between two points. The idea is to compare the distance between on digit from the other digits. The distance are put into an array matching their respected digits. I use Numpy.argsort() to give the indexes of the distances sorted from least to greatest.

2.1 K-Nearest Neighbor

I used 30 K for this assignment. A lower K gave a higher accuracy too early during the training. I used the indexes to temporally sort the data set according the distances. I took the closest K=30 digits to the working digit. The K digits I took are counted to find the which digit appears the most. That leads to a prediction of what the working digit might be.

3 Evaluation

I put the predicted results in an array to be compared to the true values from the dataset. I import metrics from sklearn to create a confusion matrix with the results. That results an 0.836 accuracy with 1,000 train images and 0.747 accuracy with 300 test images.

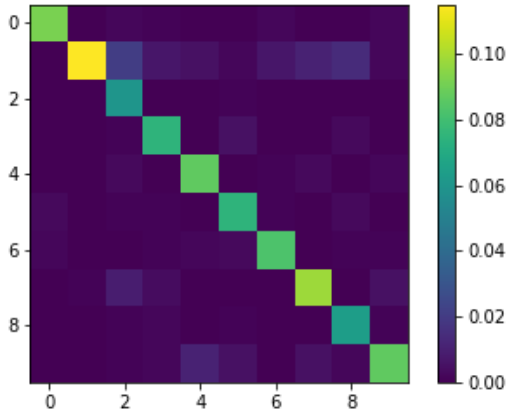

Figure 1: this Figure shows the confusion matrix of 300 test images.

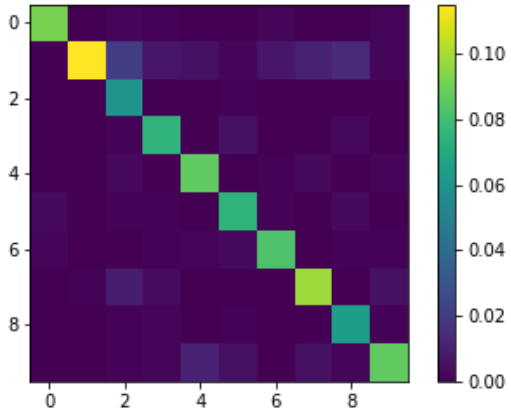

Figure 2: this Figure shows the confusion matrix of 1,000 train images.

4 Discussion

In Figure 1, we show the generated confusion matrix from the code. The colors from the top left hand corner that does down diagonally to the bottom right hand corner shows the digits that were predicted correctly. The results varies based on the amount of data I used. The more data you feed it, the accuracy rises as well. The started accuracy of 100 train digits was 0.54, then it grew to 0.836 accuracy with 1,000 train digits. Same with the test digits. 300 digits results 0.747 accuracy while 1,000 results 0.836 accuracy.

5 What I've learned

I've learned a lot from this assignment. First, is to start assignment/projects early because you will never know how long it will take to process a train set with 60,000 images. After 8 hours of execution my internet decided to give up and I never got to see the outcome. If I had started the assignment earlier, I would have not stressed too much on the wait time. I had to settle down on working with a simple for 1,000 for train and 300 for test.